R 통계 프로그래밍 패키지 제작 및 관리: Toward an Open Source Software management

2019-06-25

이남길

Summary

R 통계프로그래밍 언어를 이용하여 패키지를 제작 및 관리하는 법에 관한 지식을 정리한다.

패키지를 제작 및 관리하는 것은 그 특성이 Open Source Project와 맞닿아있다. 따라서 본 Note는 패키지의 제작, 문서화, 압축, 무결성 검사, 배포, 그리고 버전 관리 등으로 구성된다.

대부분의 내용은 Karl Broman의 “R Package Primer: a minimal tutorial”에서 발췌한다 [1].

Outline

1. Anatomy of an Open Source Project
2. 버전 관리(1): Git/Github Guide
3. 버전 관리(2): Git/Github with RStudio
4. 패키지 제작, 압축 및 배포: devtools Guide
5. 문서화: Knitr (pdf), RMarkdown (Rmd), RDocumentation (Rd)

References

[1] Karl Broman. R package primer: a minimal tutorial. <https://kbroman.org/pkg_primer/>, <https://github.com/kbroman/pkg_primer>

[2] <https://opensource.guide/how-to-contribute/>

[3] Git on Windows for Newbs. <https://code.tutsplus.com/tutorials/git-on-windows-for-newbs--net-25847>

1. Anatomy of an open source project
   1. People
      1. **Author**: The person/s or organization that created the project
      2. **Owner**: The person/s who has administrative ownership over the organization or repository (not always the same as the original author)
      3. **Maintainers**: Contributors who are responsible for driving the vision and managing the organizational aspects of the project. (They may also be authors or owners of the project.)
      4. **Contributors**: Everyone who has contributed something back to the project.
      5. **Community Members**: People who use the project. They might be active in conversations or express their opinion on the project’s direction.
   2. Files in the top level of repository
      1. **LICENSE**: By definition, every open source project must have an open source license. If the project does not have a license, it is not open source.
      2. **README**: The README is the instruction manual that welcomes new community members to the project. It explains why the project is useful and how to get started.
      3. **CONTRIBUTING**: Whereas READMEs help people use the project, contributing docs help people contribute to the project. It explains what types of contributions are needed and how the process works. While not every project has a CONTRIBUTING file, its presence signals that this is a welcoming project to contribute to.
      4. **CODE\_OF\_CONDUCT**: The code of conduct sets ground rules for participants’ behavior associated and helps to facilitate a friendly, welcoming environment. While not every project has a CODE\_OF\_CONDUCT file, its presence signals that this is a welcoming project to contribute to.
      5. **Other documentation**: There might be additional documentation, such as tutorials, walkthroughs, or governance policies, especially on bigger projects.
   3. Tools to organize discussion
      1. **LICENSE**: By definition, every open source project must have an open source license. If the project does not have a license, it is not open source.
      2. **README**: The README is the instruction manual that welcomes new community members to the project. It explains why the project is useful and how to get started.
      3. **CONTRIBUTING**: Whereas READMEs help people use the project, contributing docs help people contribute to the project. It explains what types of contributions are needed and how the process works. While not every project has a CONTRIBUTING file, its presence signals that this is a welcoming project to contribute to.
      4. **CODE\_OF\_CONDUCT**: The code of conduct sets ground rules for participants’ behavior associated and helps to facilitate a friendly, welcoming environment. While not every project has a CODE\_OF\_CONDUCT file, its presence signals that this is a welcoming project to contribute to.
      5. **Other documentation**: There might be additional documentation, such as tutorials, walkthroughs, or governance policies, especially on bigger projects.
2. 버전 관리(1): git/github Guide
   1. Why git?
      1. version control을 통한 변경 사항 관리(keep track of changes): 압축파일에 일일이 번호 매기는 것 보다, git을 이용하면 어떤 line에 변경이 생겼는지 까지도 찾을 수 있다.
      2. 동업자들이 만든 변경사항들을 통합: git merge
   2. Why github?
      1. Like Facebook for programmers
      2. Real Open source
      3. Lower barriers to collaboration
      4. Easy to set up git server
   3. Your first time with git/github

* Git on Linux or Git Bash: **by command line**

<https://kbroman.org/github_tutorial/pages/first_time.html>

* Git on Windows: **by Git GUI**

<https://code.tutsplus.com/tutorials/git-on-windows-for-newbs--net-25847>

1. **Get a [github](https://github.com/) account.**
2. **Install a code editor**
   1. Recommended editors: <https://gist.github.com/Zirak/9999e97e01a7bd0a76f6>
   2. Windows를 위해서 [Notepad++](http://notepad-plus-plus.org/)이 추천됨
   3. 편집기 예: Nano, Vim, Notepad++, Microsoft VSCode, VSCode Insiders, Sublime Text, Atom, …
3. **Download and install [git](https://git-scm.com/downloads).** 
   1. Git for windows: <https://git-scm.com/download/win>
      1. Use the default options for each step in the installation: It is important that you use all the default settings because they are supported by all major repository vendors such as Github, Bitbucket, and Beanstalk.
      2. Go to Start > All Programs > Git > Git GUI and make a Desktop Shortcut
4. **Set up git with your user name and email.**
   1. Open a terminal/shell such as Git Bash, and type

$ git config --global user.name "Your name here"

$ git config --global user.email "your\_email@example.com"

* 1. You can also do:

$ git config --global color.ui true

$ git config --global core.editor emacs

* 1. GUI를 이용한 방법: Git GUI > Edit > Options… 새로 뜨는 Options창에서 User Name과 Email Address를 수정한다.

1. **Set up ssh on your computer.** I like [Roger Peng](http://www.biostat.jhsph.edu/~rpeng)’s [guide to setting up password-less logins](http://www.biostat.jhsph.edu/bit/nopassword.html). Also see [github’s guide to generating SSH keys](https://help.github.com/articles/generating-ssh-keys). SSH Key에 대한 한글 소개: <https://opentutorials.org/module/432/3742>
   1. Setup SSH Keys for Windows
      1. Open Git GUI
      2. Now click on Show SSH Key under the Help Menu.
      3. Click “Generate Key”
      4. Passphrase를 입력한다. 이것은 일종의 비밀번호 역할을 한다. 자동로그인을 하려면 입력하지 않고 Enter를 누른다.
      5. “Copy To Clipboard”를 클릭하여 OpenSSH public Key를 복사한다.
2. **Paste your ssh public key into your github account settings.**
   1. Go to your github > Settings > SSH Keys on the left
   2. Click “New SSH key” on the right
   3. Add a label (like “My laptop”) and paste the public key into the big text box.
   4. In a terminal/shell, type the following to test it: $ ssh -T git@github.com
   5. Routine use of git and github

여기서는 Windows의 Git GUI를 이용한 방법을 기술한다. <https://code.tutsplus.com/tutorials/git-on-windows-for-newbs--net-25847>. Command line을 이용한 방법은 K. Broman의 <https://kbroman.org/github_tutorial/pages/routine.html> 을 참고하라.

* + 1. **Remote Repository 생성하기**
    2. **Local Repository 생성하기**
       1. Git GUI > Click “Create New Repository”
       2. 내 컴퓨터 local repository의 경로(location) 지정하기: 기존 폴더를 우선 검색하여 선택하고 그 아래에 생성할 새로운 폴더명을 붙어쓴다. 지정하는 폴더 이름은 기존 폴더명과 동일하면 안됨.
       3. 새로운 폴더(local repository) 생성 후 그 안에 새로운 파일을 추가하는 것도 가능하다.
       4. Github에 Remote repository를 생성하려면 local에서 파일 생성 > Commit > Push를 수행해야 한다.
    3. **Clone a Remote Repository to a Local Repository**
       1. Git GUI > Click “Clone Existing Repository”
       2. Source Location: 예: git@github.com:Blaxus/test-repo.git
       3. Target Location: 위의 ii에서 Local Repository생성과정과 유사함: 기존 폴더를 우선 검색하여 선택하고 그 아래에 생성할 새로운 폴더명을 붙여쓴다.
    4. **Working with the GUI Client**
       1. **Rescan**
          1. “Unstaged Changes” 창 안에는 새로 추가/제거/업데이트 된 파일들을 나타낸다. “Rescan”버튼을 누르면 전에 발생했던 이러한 바뀐 점들을 볼 수 있다.
       2. **Stage Changed**
          1. “Stage Changed”버튼을 누르면 모든 새로운 파일을 Git index에 추가한다.
          2. 선택된 파일만 Stage에 올려놓기 위해서는:

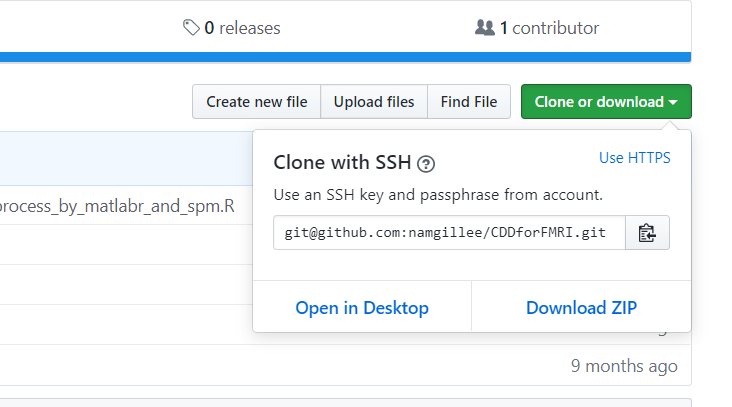
메뉴 > Commit > Stage to Commit (Ctrl-T)

* + - 1. **Commit**
         1. 모든 변경사항을 Stage한 후, 이것들을 **local** repository에 commit해야한다.
         2. 변경사항에 대한 납득할만한 Commit 메시지를 작성하시오.
         3. “Commit”버튼을 클릭하시오.
      2. **Add Remote & Push**
         1. 다른 사람들이 내 코드에 접근할 수 있도록, 이런 변경사항들을 hosted repository에 Push하여야한다.
         2. Git GUI > Remote > Add…: host repository를 지정한다.

A remote name, for example, origin

A remote URL, for example, git@github.com:user/repo.git

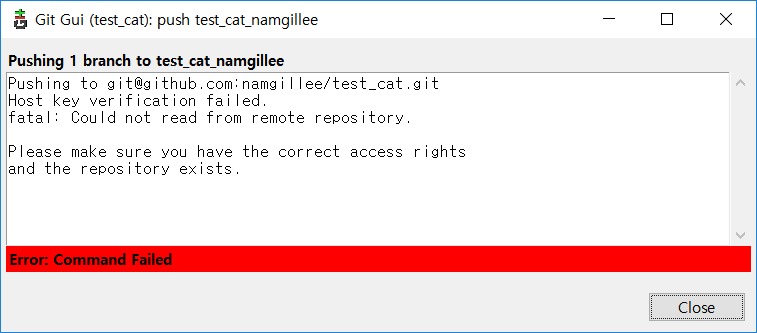
\* Github의 특정 repository의 remote URL은 아래와 같이 찾을 수 있다: 원하는 repository에 들어가서 “Clone or download”버튼을 누른 후, “USE SSH” 또는 “USE HTTPS” 버튼을 클릭하여 변경하면 확인가능하다.



* + - * 1. Git GUI > Remote > Push…
        2. 주의사항

local에 repository의 변경사항이 생긴 시기 보다 remote repository에 변경사항이 생긴 시기가 더 최신인 경우, Push가 불가능하다. 이 경우, Pull, 즉, remote로부터 Remote > Fetch from… 및 Merge > Local Merge… 를 수행해야 한다. (이것은 보고서의 다음 섹션에 추가해야 한다.)

remote repository를 생성하지 않았다면 에러가 발생함. 즉, Push는 Remote repository를 생성하는 작동이 아니다.



test\_cat이라는 repository가 존재하지 않기 때문에 위와 같은 메시지가 뜬다.

* 1. .gitignore
     1. The various files in your project directory that you’re not tracking in git should be indicated in a .gitignore file. You don’t have to have a .gitignore file, but if you don’t, those files will show up every time you type git status.
     2. Each subdirectory can have its own .gitignore file, too.
     3. Also, you can have a global such in your home directory; I use ~/.gitignore\_global, which contains:

\*~

.\*~

.DS\_Store

.Rhistory

.RData

* + 1. You have to tell git about the global .gitignore file:

$ git config --global core.excludesfile ~/.gitignore\_global

1. 버전 관리(2): Git/Github with RStudio

[1] Karl Broman. <https://kbroman.org/github_tutorial/pages/rstudio.html>.

[2] Nathan Stephens. <https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN>.